4th International Workshop on
Interactions between Computer Science and Biology

Under-approximation of Reachability in Multivalued Asynchronous Networks

Maxime FOLSCHETTE

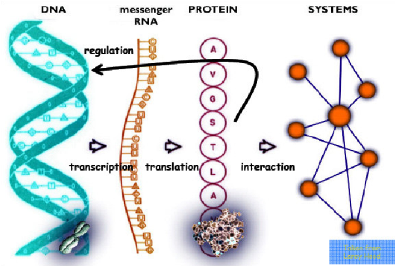
MeForBio / IRCCyN / École Centrale de Nantes (Nantes, France)
maxime.folschette@irccyn.ec-nantes.fr
http://www.irccyn.ec-nantes.fr/~folschet/

Joint work with:
Loïc PAULEVÉ, Morgan MAGNIN, Olivier ROUX

Context and Aims

MeForBio team:

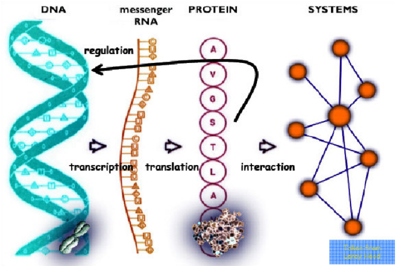
Algebraic modelling to study complex dynamical biological systems



Context and Aims

MeForBio team:

Algebraic modelling to study complex dynamical biological systems

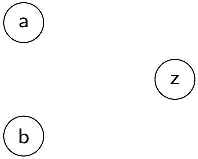


1) Asynchronous Discrete Networks (ADN)
   Convenient to model biological systems
2) Process Hitting (PH)
   Cannot accurately describe ADNs
3) Enhancing PH with priorities
   To efficiently compute reachability in ADNs

The Asynchronous Discrete Networks (ADN)

[De Jong in *Journal of Computational Biology*, 2002]

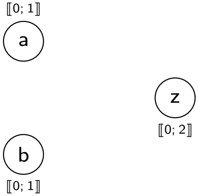
- A set of components $N = \{a, b, z\}$



The Asynchronous Discrete Networks (ADN)

[De Jong in *Journal of Computational Biology*, 2002]

- A set of components $N = \{a, b, z\}$
- A set of expression levels for each component $z \in \mathbb{F}^z = [\![0; 2]\!]$
- The set of global states $\mathbb{F} = \mathbb{F}^a \times \mathbb{F}^b \times \mathbb{F}^z$



The Asynchronous Discrete Networks (ADN)

[De Jong in *Journal of Computational Biology*, 2002]

- A set of components $N = \{a, b, z\}$
- A set of expression levels for each component $z \in \mathbb{F}^z = [\![0; 2]\!]$
- The set of global states $\mathbb{F} = \mathbb{F}^a \times \mathbb{F}^b \times \mathbb{F}^z$
- An evolution function for each component $f^z : \mathbb{F} \to \mathbb{F}^z$

$f^a = \neg b$

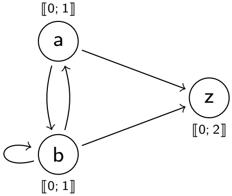
b	$f^a(b)$
0	**1**
1	**0**

$f^b = b \vee \neg a$

a	b	$f^b(a,b)$
0	0	**1**
0	1	**1**
1	0	**0**
1	1	**1**

$f^z = a + b$

a	b	$f^z(a,b)$
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**2**



The Asynchronous Discrete Networks (ADN)

State Graph: $\mathrm{G} = (\mathbb{F}, \mathbb{E})$, where one component evolves at a time given its function f^a

$$(x, y) \in \mathbb{E} \iff \exists a \in N, y^a = f^a(x) \land \forall b \neq a, y^b = x^b$$

The Asynchronous Discrete Networks (ADN)

State Graph: $\mathrm{G} = (\mathbb{F}, \mathbb{E})$, where one component evolves at a time given its function f^a

$$(x, y) \in \mathbb{E} \Longleftrightarrow \exists a \in N, y^a = f^a(x) \land \forall b \neq a, y^b = x^b$$

Size of the State Graph: $|\mathbb{F}| = \prod_{a \in N} |\mathbb{F}^a| \quad \geq 2^{|N|}$

$\rightarrow$ **Exponential** in the number $|N|$ of components

The Asynchronous Discrete Networks (ADN)

State Graph: $\mathrm{G} = (\mathbb{F}, \mathbb{E})$, where one component evolves at a time given its function f^a

$$(x, y) \in \mathbb{E} \iff \exists a \in N, y^a = f^a(x) \land \forall b \neq a, y^b = x^b$$

Size of the State Graph: $|\mathbb{F}| = \prod_{a \in N} |\mathbb{F}^a| \quad \geq 2^{|N|}$

$\rightarrow$ **Exponential** in the number $|N|$ of components

Some works give a link between the structure and the behaviour of an ADN

- **Thomas' conjecture** (condition for multiple fixed points or attractive cycle)
  - Boolean: [Remy, Ruet, Thieffry in *Advances in Applied Mathematics*, 2008]
  - Multivalued: [Richard, Comet in *Discrete Applied Mathematics*, 2007]

The Asynchronous Discrete Networks (ADN)

State Graph: $\mathrm{G} = (\mathbb{F}, \mathbb{E})$, where one component evolves at a time given its function f^a

$$(x, y) \in \mathbb{E} \iff \exists a \in N, y^a = f^a(x) \land \forall b \neq a, y^b = x^b$$

Size of the State Graph: $|\mathbb{F}| = \prod_{a \in N} |\mathbb{F}^a| \quad \geq 2^{|N|}$

$\rightarrow$ **Exponential** in the number $|N|$ of components

Some works give a link between the structure and the behaviour of an ADN

- **Thomas' conjecture** (condition for multiple fixed points or attractive cycle)
  - Boolean: [Remy, Ruet, Thieffry in *Advances in Applied Mathematics*, 2008]
  - Multivalued: [Richard, Comet in *Discrete Applied Mathematics*, 2007]

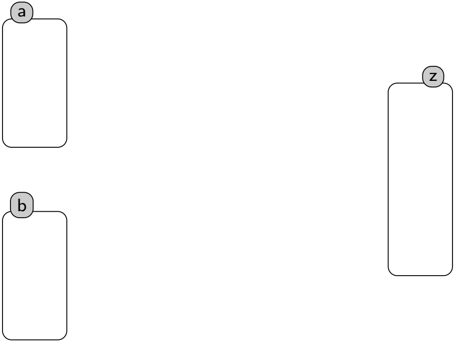
But methods related to reachability rely on the State Graph

e.g.: Starting from $(a, b, z) = (0, 0, 0)$, can the system reach $z = 2$?

- **Temporal logics**
  - CTL: [Bernot, Comet, Richard, Guespin in *Journal of Theoretical Biology*, 2004]
  - LTL: [Ito, Izumi, Hagihara, Yonezaki in *BioInformatics and BioEngineering*, 2010]

The Process Hitting modeling

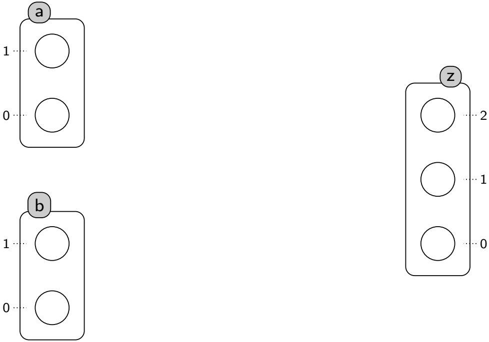
[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology*, 2011]



Sorts: components a, b, z

The Process Hitting modeling

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology*, 2011]

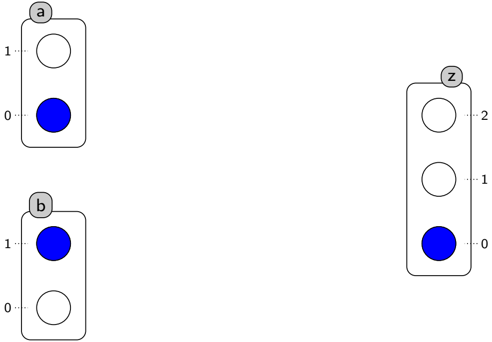


Sorts: components a, b, z

Processes: local states / levels of expression z_0, z_1, z_2

The Process Hitting modeling

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology*, 2011]



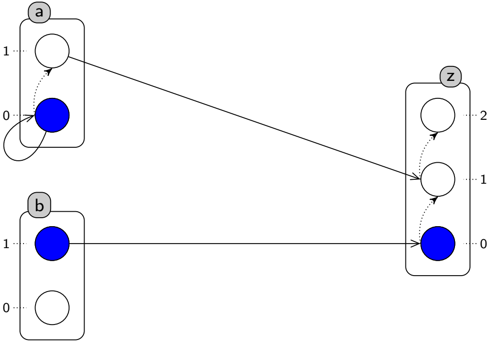
Sorts: components a, b, z

Processes: local states / levels of expression z_0, z_1, z_2

States: sets of active processes $\langle a_0, b_1, z_0 \rangle$

The Process Hitting modeling

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology*, 2011]



Sorts: components a, b, z

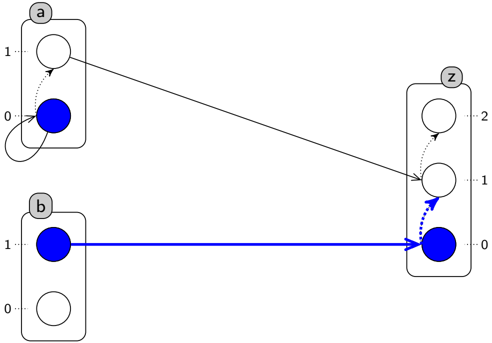
Processes: local states / levels of expression z_0, z_1, z_2

States: sets of active processes $\langle a_0, b_1, z_0 \rangle$

Actions: dynamics $b_1 \rightarrow z_0 \upharpoonright z_1,\ a_0 \rightarrow a_0 \upharpoonright a_1,\ a_1 \rightarrow z_1 \upharpoonright z_2$

The Process Hitting modeling

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology*, 2011]



Sorts: components a, b, z

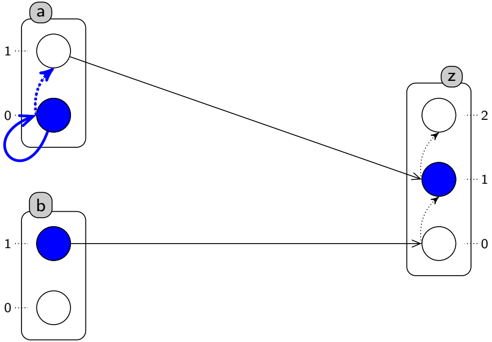
Processes: local states / levels of expression z_0, z_1, z_2

States: sets of active processes $\langle a_0, b_1, z_0 \rangle$

Actions: dynamics $\underline{b_1 \rightarrow z_0 \Rsh z_1}$, $a_0 \rightarrow a_0 \Rsh a_1$, $a_1 \rightarrow z_1 \Rsh z_2$

The Process Hitting modeling

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology*, 2011]



Sorts: components a, b, z

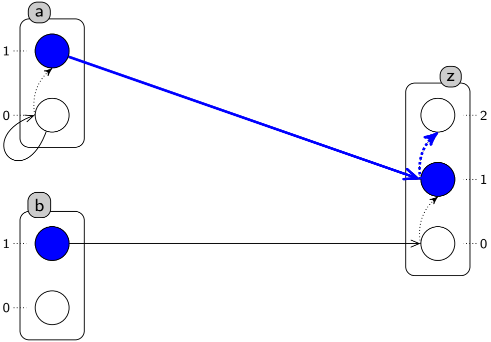
Processes: local states / levels of expression z_0, z_1, z_2

States: sets of active processes $\langle a_0, b_1, z_1 \rangle$

Actions: dynamics $b_1 \rightarrow z_0 \upharpoonright z_1,\ \underline{a_0 \rightarrow a_0 \upharpoonright a_1},\ a_1 \rightarrow z_1 \upharpoonright z_2$

The Process Hitting modeling

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology*, 2011]



Sorts: components $a,\ b,\ z$

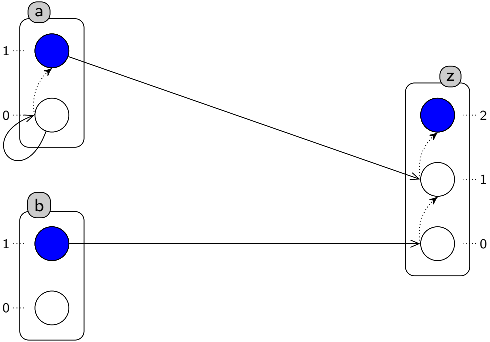
Processes: local states / levels of expression $z_0,\ z_1,\ z_2$

States: sets of active processes $\langle a_1, b_1, z_1 \rangle$

Actions: dynamics $b_1 \rightarrow z_0 \upharpoonright z_1,\ a_0 \rightarrow a_0 \upharpoonright a_1,\ \underline{a_1 \rightarrow z_1 \upharpoonright z_2}$

The Process Hitting modeling

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology*, 2011]



Sorts: components a, b, z

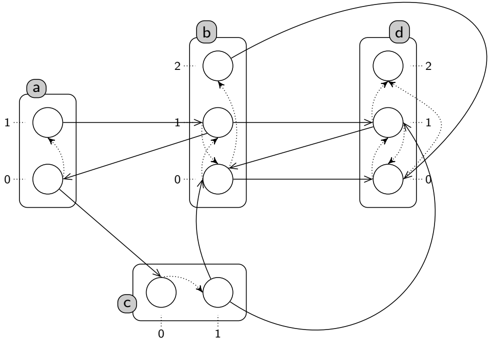
Processes: local states / levels of expression z_0, z_1, z_2

States: sets of active processes $\langle a_1, b_1, z_2 \rangle$

Actions: dynamics $b_1 \rightarrow z_0 \upharpoonright z_1$, $a_0 \rightarrow a_0 \upharpoonright a_1$, $a_1 \rightarrow z_1 \upharpoonright z_2$

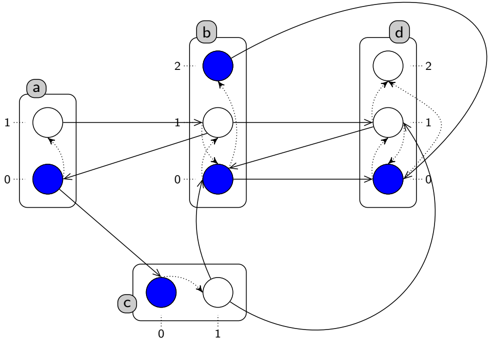
Static analysis: successive reachability of processes

[Paulevé, Magnin, Roux in *Mathematical Structures in Computer Science*, 2012]



Static analysis: successive reachability of processes

[Paulevé, Magnin, Roux in *Mathematical Structures in Computer Science*, 2012]

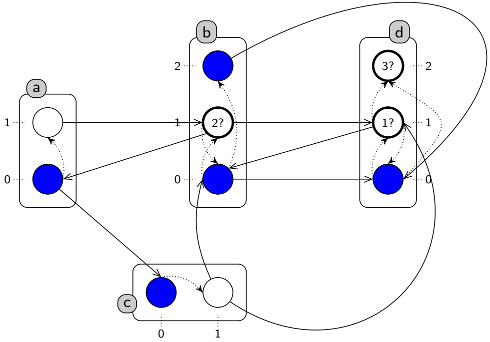


- Initial context

$\langle a_1, \{b_0, b_1\}, c_0, d_0 \rangle$

Static analysis: successive reachability of processes

[Paulevé, Magnin, Roux in *Mathematical Structures in Computer Science*, 2012]



- Initial context

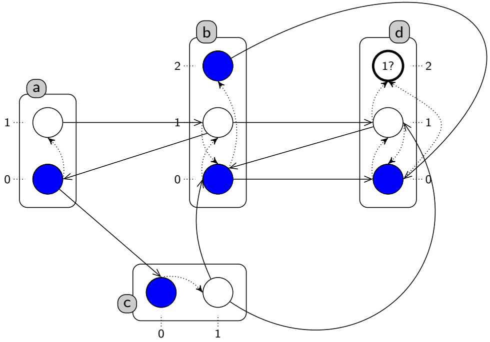
  $\langle a_1, \{b_0, b_1\}, c_0, d_0 \rangle$

- Objectives

  $[\ \Rsh d_1 :: \Rsh b_1 :: \Rsh d_2\ ]$

Static analysis: successive reachability of processes

[Paulevé, Magnin, Roux in *Mathematical Structures in Computer Science*, 2012]



- Initial context

  $\langle a_1, \{b_0, b_1\}, c_0, d_0 \rangle$

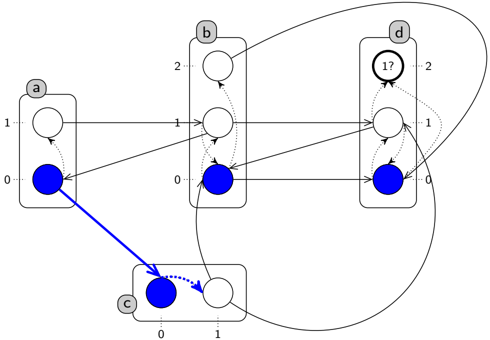
- Objectives

  $[\ \Rsh d_1 :: \Rsh b_1 :: \Rsh d_2\ ]$

  $[\ \Rsh d_2\ ]$

Static analysis: successive reachability of processes

[Paulevé, Magnin, Roux in *Mathematical Structures in Computer Science*, 2012]



- Initial context

$\langle a_1, \{b_0, b_1\}, c_0, d_0 \rangle$

- Objectives

$[\ \Rsh d_1 :: \Rsh b_1 :: \Rsh d_2\ ]$

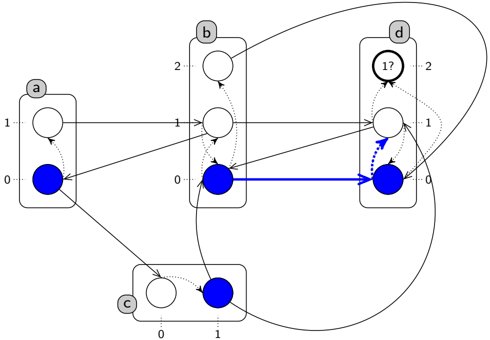
$[\ \Rsh d_2\ ]$

$\longrightarrow$ Concretization of the objective = scenario

$\underline{a_0 \to c_0 \Rsh c_1} :: b_0 \to d_0 \Rsh d_1 :: c_1 \to b_0 \Rsh b_1 :: b_1 \to d_1 \Rsh d_2$

Static analysis: successive reachability of processes

[Paulevé, Magnin, Roux in *Mathematical Structures in Computer Science*, 2012]



- Initial context

$\langle a_1, \{b_0, b_1\}, c_0, d_0\rangle$

- Objectives

$[\ \Rsh d_1 :: \Rsh b_1 :: \Rsh d_2\ ]$

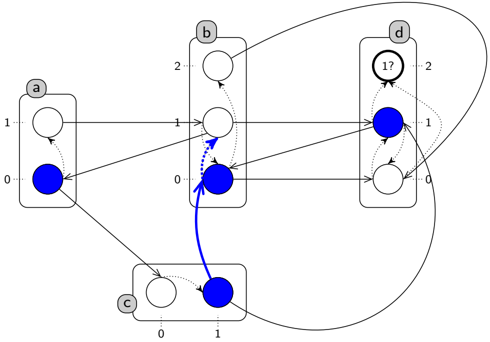
$[\ \Rsh d_2\ ]$

$\rightarrow$ Concretization of the objective = scenario

$a_0 \rightarrow c_0 \Rsh c_1 :: \underline{b_0 \rightarrow d_0 \Rsh d_1} :: c_1 \rightarrow b_0 \Rsh b_1 :: b_1 \rightarrow d_1 \Rsh d_2$

Static analysis: successive reachability of processes

[Paulevé, Magnin, Roux in *Mathematical Structures in Computer Science*, 2012]



- Initial context

$\langle a_1, \{b_0, b_1\}, c_0, d_0 \rangle$

- Objectives

$[\ \upharpoonright d_1 :: \upharpoonright b_1 :: \upharpoonright d_2\ ]$

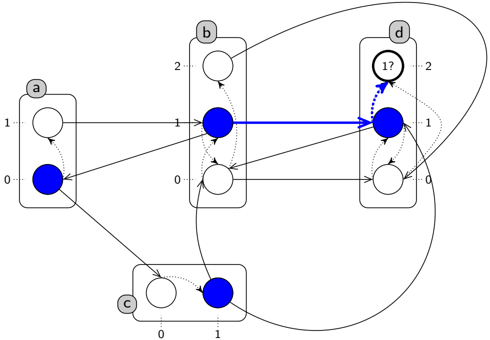
$[\ \upharpoonright d_2\ ]$

$\rightarrow$ Concretization of the objective = scenario

$a_0 \rightarrow c_0 \upharpoonright c_1 :: b_0 \rightarrow d_0 \upharpoonright d_1 :: \underline{c_1 \rightarrow b_0 \upharpoonright b_1} :: b_1 \rightarrow d_1 \upharpoonright d_2$

Static analysis: successive reachability of processes

[Paulevé, Magnin, Roux in *Mathematical Structures in Computer Science*, 2012]



- Initial context

  $\langle a_1, \{b_0, b_1\}, c_0, d_0 \rangle$

- Objectives

  $[\ \Rsh d_1 :: \Rsh b_1 :: \Rsh d_2\ ]$

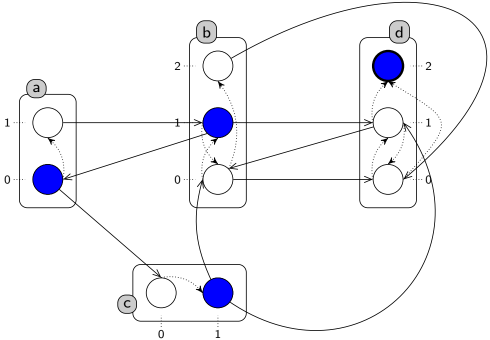
  $[\ \Rsh d_2\ ]$

$\rightarrow$ Concretization of the objective = scenario

$a_0 \rightarrow c_0 \Rsh c_1 :: b_0 \rightarrow d_0 \Rsh d_1 :: c_1 \rightarrow b_0 \Rsh b_1 :: \underline{b_1 \rightarrow d_1 \Rsh d_2}$

Static analysis: successive reachability of processes

[Paulevé, Magnin, Roux in *Mathematical Structures in Computer Science*, 2012]



- Initial context

  $\langle a_1, \{b_0, b_1\}, c_0, d_0 \rangle$

- Objectives

  $[\, \Rsh d_1 :: \Rsh b_1 :: \Rsh d_2 \,]$

  $[\, \Rsh d_2 \,]$

$\rightarrow$ Concretization of the objective = scenario

$a_0 \rightarrow c_0 \Rsh c_1 :: b_0 \rightarrow d_0 \Rsh d_1 :: c_1 \rightarrow b_0 \Rsh b_1 :: b_1 \rightarrow d_1 \Rsh d_2$

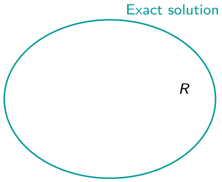
Over- and Under-approximations

[Paulevé, Magnin, Roux in *Mathematical Structures in Computer Science*, 2012]

Static analysis by abstractions:

→ Directly checking an objective sequence R is hard (**State Graph**)

→ Rather check the approximations P and Q, where $P \Rightarrow R \Rightarrow Q$:



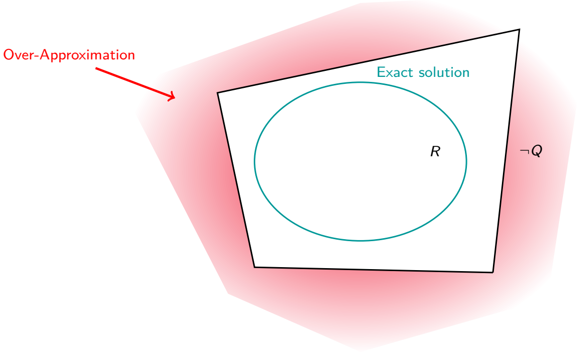
Over- and Under-approximations

[Paulevé, Magnin, Roux in *Mathematical Structures in Computer Science*, 2012]

Static analysis by abstractions:

$\rightarrow$ Directly checking an objective sequence R is hard (**State Graph**)

$\rightarrow$ Rather check the approximations P and Q, where $P \Rightarrow R \Rightarrow Q$:



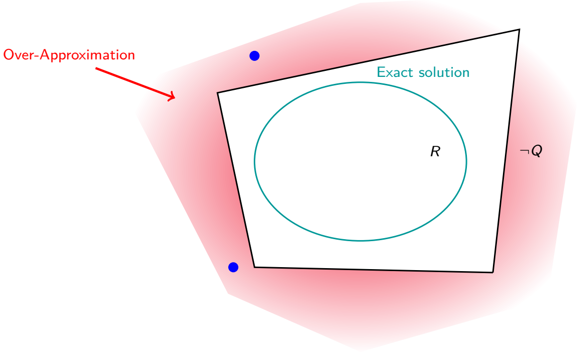
Over- and Under-approximations

[Paulevé, Magnin, Roux in *Mathematical Structures in Computer Science*, 2012]

Static analysis by abstractions:

→ Directly checking an objective sequence R is hard (**State Graph**)

→ Rather check the approximations P and Q, where $P \Rightarrow R \Rightarrow Q$:



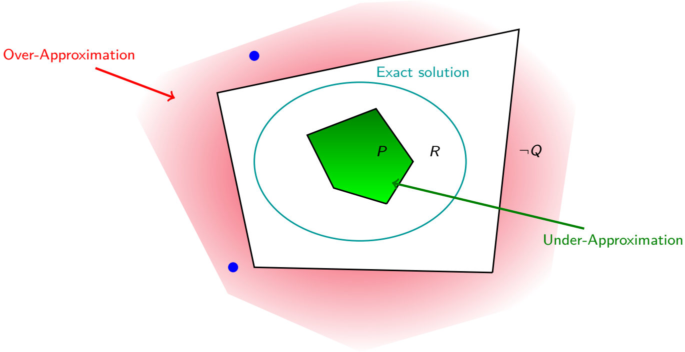
Over- and Under-approximations

[Paulevé, Magnin, Roux in *Mathematical Structures in Computer Science*, 2012]

Static analysis by abstractions:

→ Directly checking an objective sequence R is hard (**State Graph**)

→ Rather check the approximations P and Q, where $P \Rightarrow R \Rightarrow Q$:



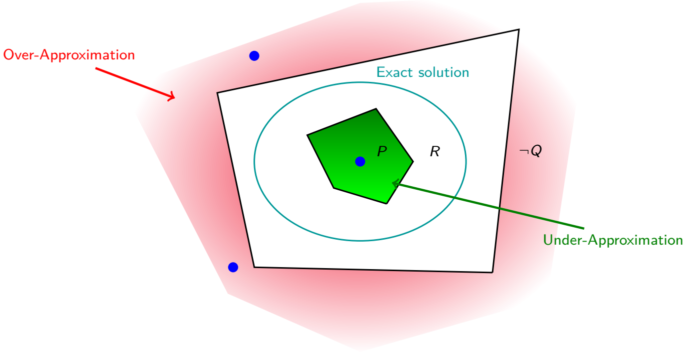
Over- and Under-approximations

[Paulevé, Magnin, Roux in *Mathematical Structures in Computer Science*, 2012]

Static analysis by abstractions:

→ Directly checking an objective sequence R is hard (**State Graph**)

→ Rather check the approximations P and Q, where $P \Rightarrow R \Rightarrow Q$:



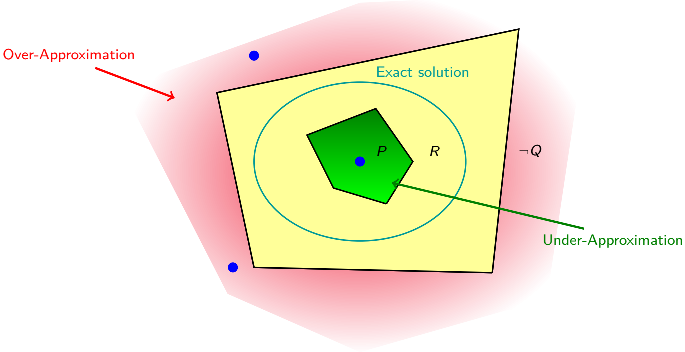
Over- and Under-approximations

[Paulevé, Magnin, Roux in *Mathematical Structures in Computer Science*, 2012]

Static analysis by abstractions:

→ Directly checking an objective sequence R is hard (**State Graph**)

→ Rather check the approximations P and Q, where $P \Rightarrow R \Rightarrow Q$:



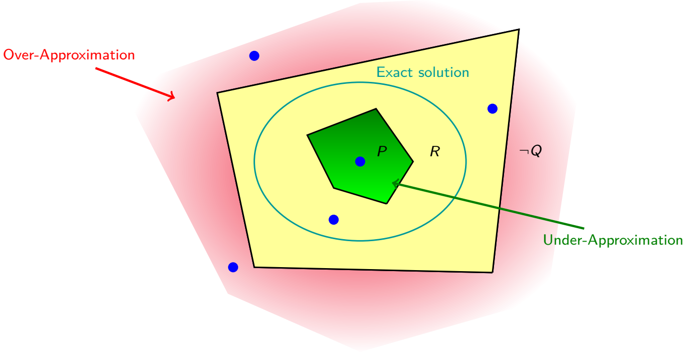
Over- and Under-approximations

[Paulevé, Magnin, Roux in *Mathematical Structures in Computer Science*, 2012]

Static analysis by abstractions:

$\longrightarrow$ Directly checking an objective sequence R is hard (**State Graph**)

$\longrightarrow$ Rather check the approximations P and Q, where $P \Rightarrow R \Rightarrow Q$:



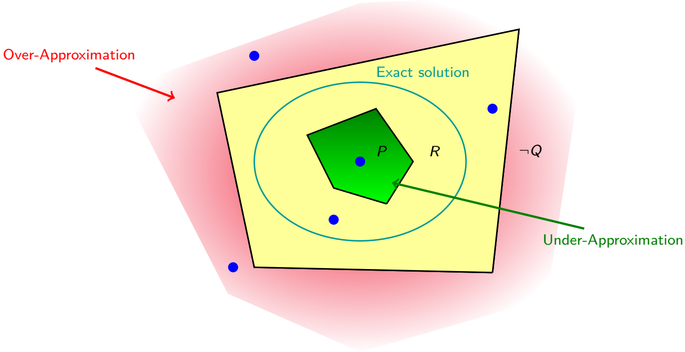
Over- and Under-approximations

[Paulevé, Magnin, Roux in *Mathematical Structures in Computer Science*, 2012]

Static analysis by abstractions:

→ Directly checking an objective sequence R is hard (**State Graph**)

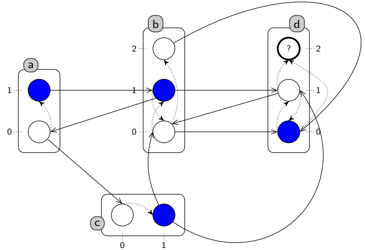
→ Rather check the approximations P and Q, where $P \Rightarrow R \Rightarrow Q$:



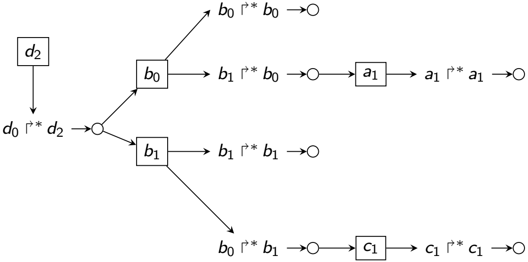
Computing P or Q is **polynomial** in the number of **sorts** and **exponential** in the number of **processes in each sort**

→ Efficient for big models with few levels of expression

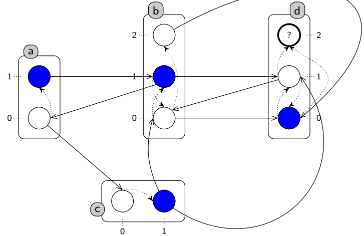
Under-approximation



$\boxed{d_2}$	Required process
$d_0 \Rsh^* d_2$	Objective
○	Solution to an objective



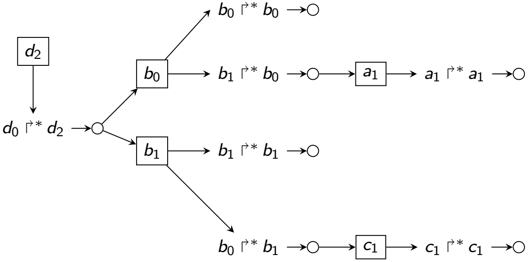
Under-approximation



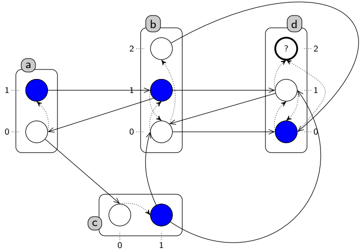
Sufficient condition:

- no cycle
- each objective has a solution

$\boxed{d_2}$	Required process
$d_0 \Rsh^* d_2$	Objective
○	Solution to an objective



Under-approximation

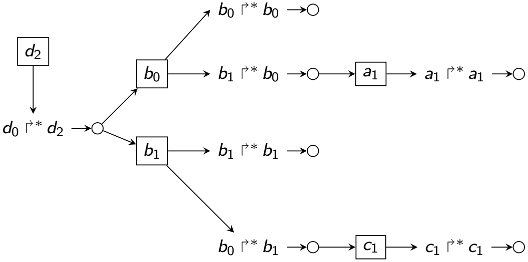


Sufficient condition:

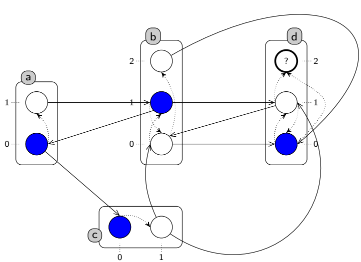
- no cycle
- each objective has a solution

R is **true**

$\boxed{d_2}$	Required process
$d_0 \Rsh^* d_2$	Objective
○	Solution to an objective

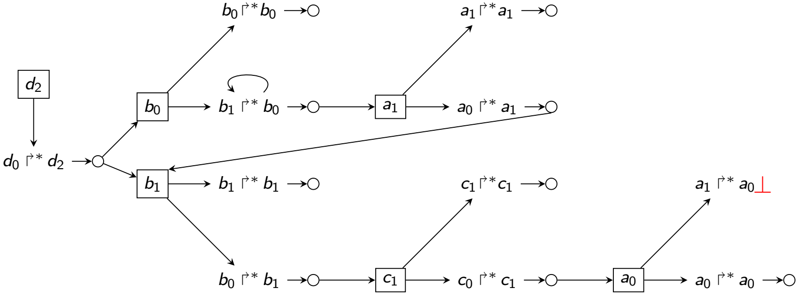


Under-approximation

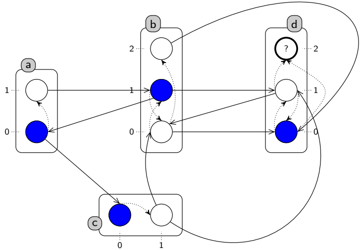


Sufficient condition:

- no cycle
- ~~each objective has a solution~~



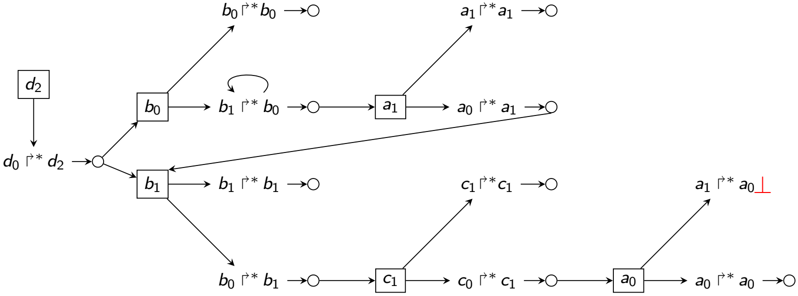
Under-approximation



Sufficient condition:

- no cycle
- ~~each objective has a solution~~

Inconclusive



Implementation in PINT

Existing free OCaml library: PINT

→ Compiler + tools for Process Hitting models
→ Documentation & examples: http://processhitting.wordpress.com/

Implementation in PINT

Existing free OCaml library: PINT

→ Compiler + tools for Process Hitting models
→ Documentation & examples: http://processhitting.wordpress.com/

Computation time for various reachability analyses:

Model	Sorts	Procs	Actions	States	Biocham[1]	libddd[2]	PINT
egfr20	35	196	670	2^{64}	[3s – ∞]	[1s – 150s]	**0.007s**
tcrsig40	54	156	301	2^{73}	[1s – ∞]	[0.6s – ∞]	**0.004s**
tcrsig94	133	448	1124	2^{194}	∞	∞	**0.030s**
egfr104	193	748	2356	2^{320}	∞	∞	**0.050s**

[1] Inria Paris-Rocquencourt/Contraintes
[2] LIP6/Move

egfr20: [Epidermal Growth Factor Receptor, by Özgür Sahin *et al.*]
egfr104: [Epidermal Growth Factor Receptor, by Regina Samaga *et al.*]
tcrsig40: [T-Cell Receptor Signaling, by Steffen Klamt *et al.*]
tcrsig94: [T-Cell Receptor Signaling, by Julio Saez-Rodriguez *et al.*]

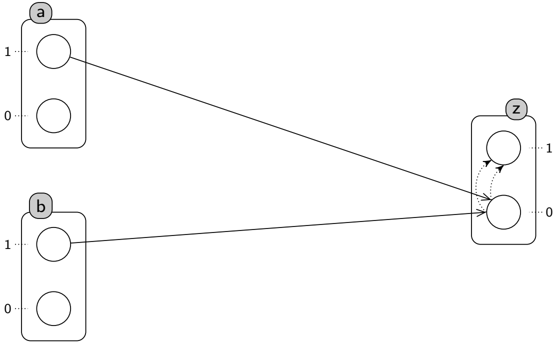
Adding cooperations

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology*, 2011]

Cooperation between a_1 and b_1: $\underline{a_1 \wedge b_1} \rightarrow z_0 \upharpoonright z_1$

Adding cooperations

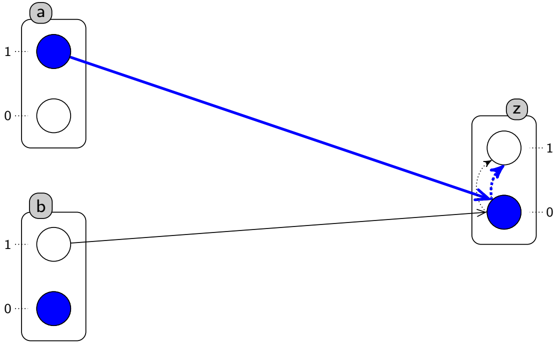
[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology*, 2011]



Cooperation between a_1 and b_1: $\underline{a_1 \wedge b_1} \rightarrow z_0 \Rsh z_1$

Adding cooperations

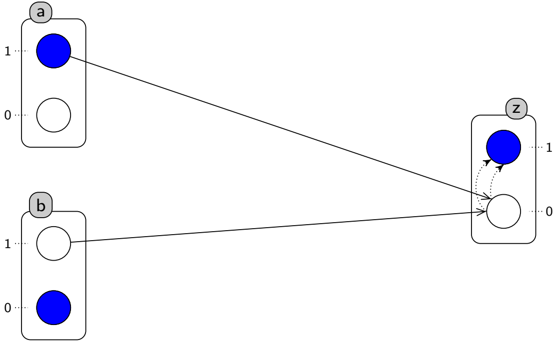
[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology*, 2011]



Cooperation between a_1 and b_1: $\underline{a_1 \wedge b_1} \rightarrow z_0 \upharpoonright z_1$

Adding cooperations

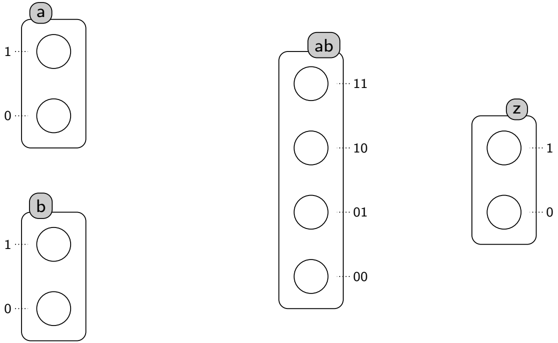
[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology*, 2011]



Cooperation between a_1 and b_1: $\underline{a_1 \wedge b_1} \rightarrow z_0 \upharpoonright z_1$

Adding cooperations

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology*, 2011]

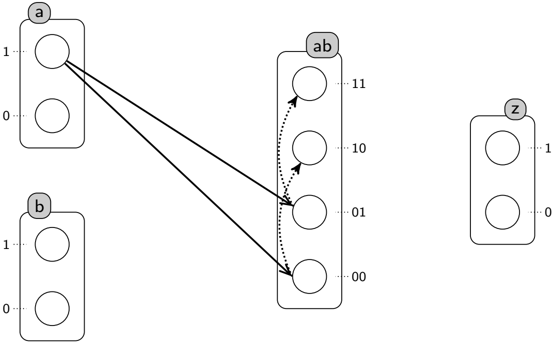


Cooperation between a_1 and b_1: $\underline{a_1 \wedge b_1} \rightarrow z_0 \upharpoonright z_1$

Solution: a **cooperative sort** ab to express $\underline{a_1 \wedge b_1}$

Adding cooperations

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology*, 2011]

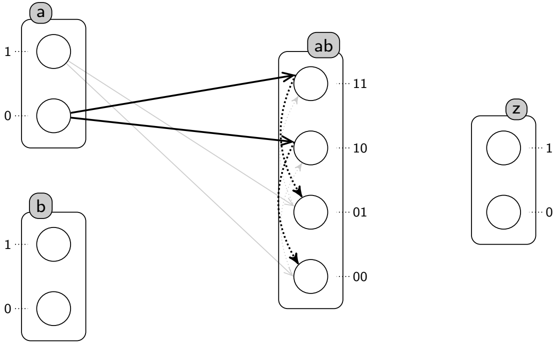


Cooperation between a_1 and b_1: $\underline{a_1 \wedge b_1} \to z_0 \Rsh z_1$

Solution: a **cooperative sort** ab to express $\underline{a_1 \wedge b_1}$

Adding cooperations

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology*, 2011]

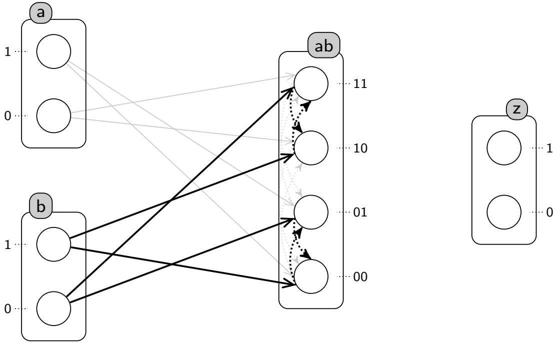


Cooperation between a_1 and b_1: $\underline{a_1 \wedge b_1} \rightarrow z_0 \Rsh z_1$

Solution: a **cooperative sort** ab to express $\underline{a_1 \wedge b_1}$

Adding cooperations

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology*, 2011]

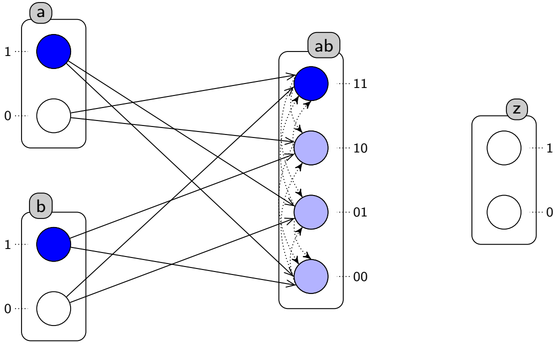


Cooperation between a_1 and b_1: $\underline{a_1 \wedge b_1} \rightarrow z_0 \upharpoonright z_1$

Solution: a **cooperative sort** ab to express $\underline{a_1 \wedge b_1}$

Adding cooperations

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology*, 2011]



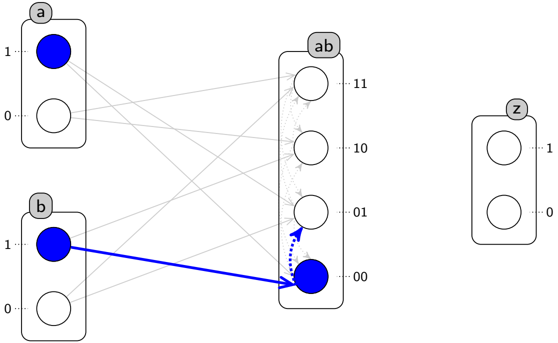
Cooperation between a_1 and b_1: $\underline{a_1 \wedge b_1} \rightarrow z_0 \upharpoonright z_1$

Solution: a **cooperative sort** ab to express $\underline{a_1 \wedge b_1}$

Constraint: each configuration is represented by one process $\underline{a_1 \wedge b_1} \Rightarrow ab_{11}$

Adding cooperations

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology*, 2011]



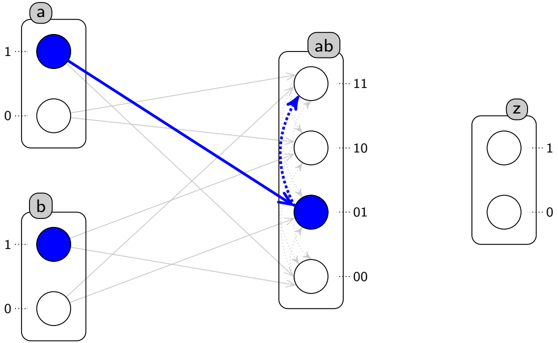
Cooperation between a_1 and b_1: $\underline{a_1 \wedge b_1} \rightarrow z_0 \upharpoonright z_1$

Solution: a **cooperative sort** ab to express $\underline{a_1 \wedge b_1}$

Constraint: each configuration is represented by one process $\underline{a_1 \wedge b_1} \Rightarrow ab_{11}$

Adding cooperations

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology*, 2011]



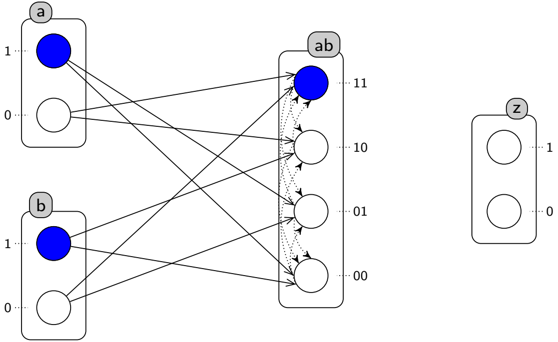
Cooperation between a_1 and b_1: $\underline{a_1 \wedge b_1} \rightarrow z_0 \upharpoonright z_1$

Solution: a **cooperative sort** ab to express $\underline{a_1 \wedge b_1}$

Constraint: each configuration is represented by one process $\underline{a_1 \wedge b_1} \Rightarrow ab_{11}$

Adding cooperations

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology*, 2011]



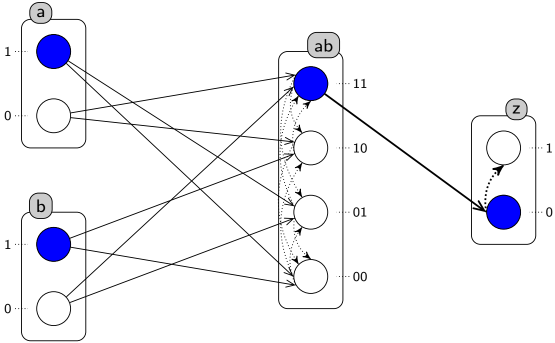
Cooperation between a_1 and b_1: $\underline{a_1 \wedge b_1} \rightarrow z_0 \upharpoonright z_1$

Solution: a **cooperative sort** ab to express $\underline{a_1 \wedge b_1}$

Constraint: each configuration is represented by one process $\underline{a_1 \wedge b_1} \Rightarrow ab_{11}$

Adding cooperations

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology*, 2011]



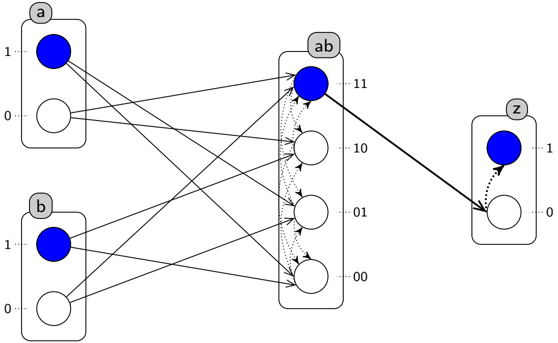
Cooperation between a_1 and b_1: $\underline{a_1 \wedge b_1} \to z_0 \upharpoonright z_1$

Solution: a **cooperative sort** ab to express $\underline{a_1 \wedge b_1}$

Constraint: each configuration is represented by one process $\underline{a_1 \wedge b_1} \Rightarrow ab_{11}$

Adding cooperations

[Paulevé, Magnin, Roux in *Transactions on Computational Systems Biology*, 2011]

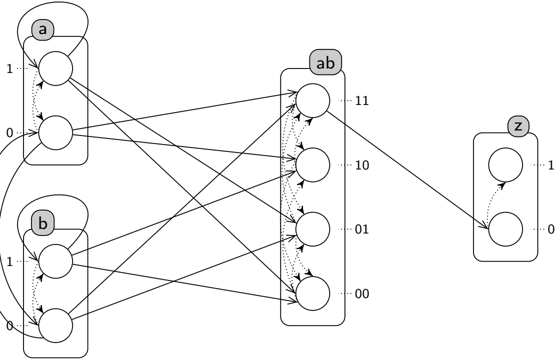


Cooperation between a_1 and b_1: $\underline{a_1 \wedge b_1} \rightarrow z_0 \upharpoonright z_1$

Solution: a **cooperative sort** ab to express $\underline{a_1 \wedge b_1}$

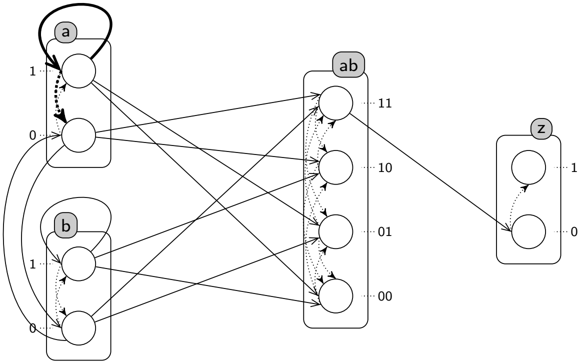
Constraint: each configuration is represented by one process $\underline{a_1 \wedge b_1} \Rightarrow ab_{11}$

Adapting the expressivity of PH



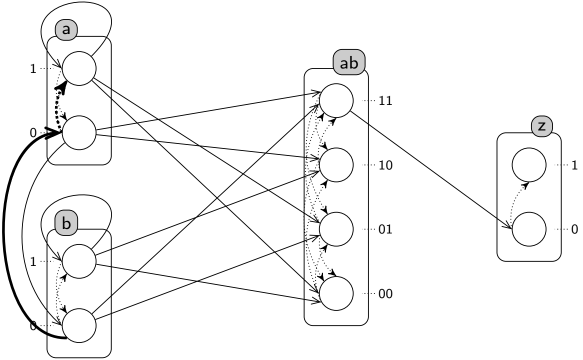
Drawback: Cooperations are too “loose” to be as expressive as ADN.

Adapting the expressivity of PH



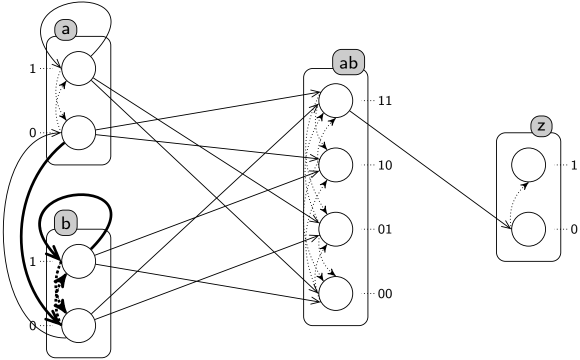
Drawback: Cooperations are too “loose” to be as expressive as ADN.

Adapting the expressivity of PH



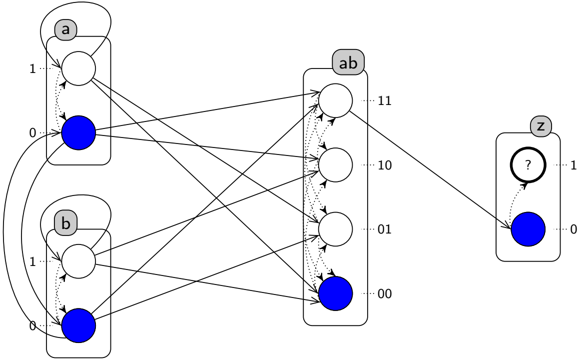
Drawback: Cooperations are too “loose” to be as expressive as ADN.

Adapting the expressivity of PH



Drawback: Cooperations are too "loose" to be as expressive as ADN.

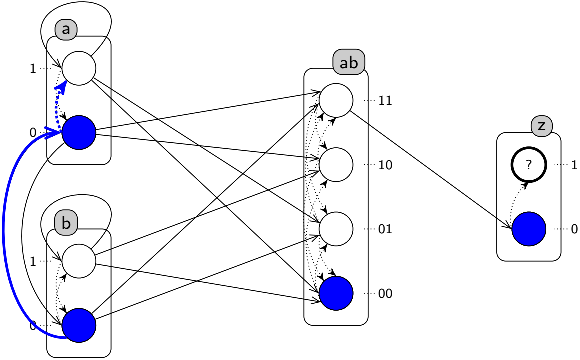
Adapting the expressivity of PH



Drawback: Cooperations are too "loose" to be as expressive as ADN.

$\langle a_0, b_0, ab_{00}, z_0 \rangle$

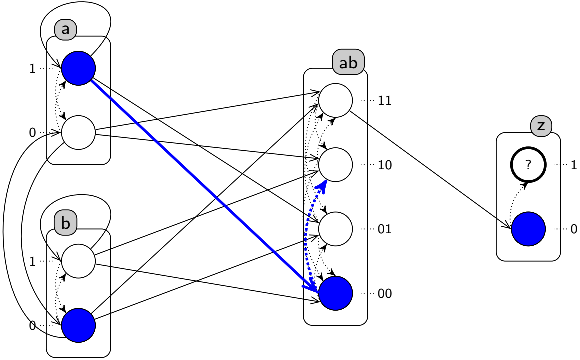
Adapting the expressivity of PH



Drawback: Cooperations are too “loose” to be as expressive as ADN.

$\langle a_0, b_0, ab_{00}, z_0 \rangle$

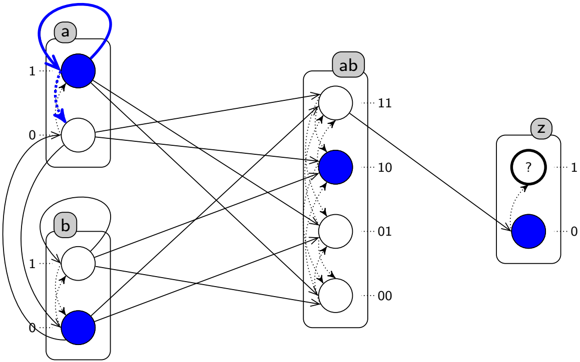
Adapting the expressivity of PH



Drawback: Cooperations are too “loose” to be as expressive as ADN.

$\langle a_0, b_0, ab_{00}, z_0\rangle \rightarrow \langle a_1, b_0, ab_{00}, z_0\rangle$

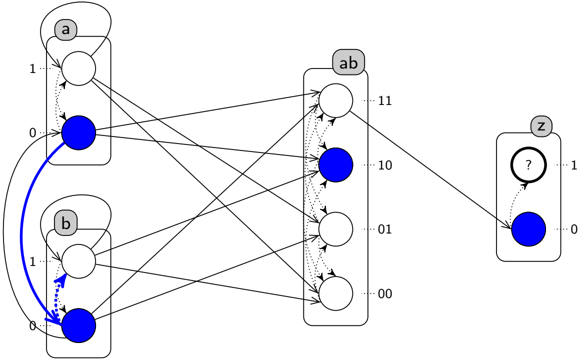
Adapting the expressivity of PH



Drawback: Cooperations are too "loose" to be as expressive as ADN.

$\langle a_0, b_0, ab_{00}, z_0 \rangle \rightarrow \langle a_1, b_0, ab_{00}, z_0 \rangle \rightarrow \langle a_1, b_0, ab_{10}, z_0 \rangle$

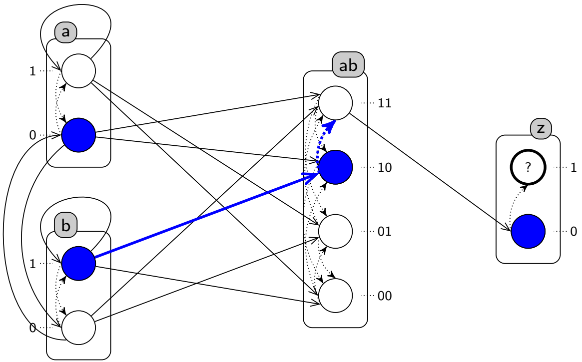
Adapting the expressivity of PH



Drawback: Cooperations are too “loose” to be as expressive as ADN.

$\langle a_0, b_0, ab_{00}, z_0 \rangle \rightarrow \langle a_1, b_0, ab_{00}, z_0 \rangle \rightarrow \langle a_1, b_0, ab_{10}, z_0 \rangle \rightarrow \langle a_0, b_0, ab_{10}, z_0 \rangle$

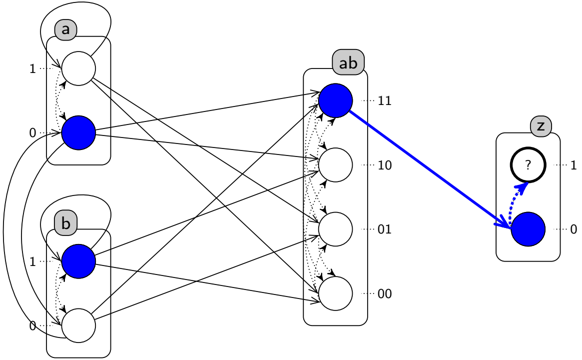
Adapting the expressivity of PH



Drawback: Cooperations are too “loose” to be as expressive as ADN.

$$\langle a_0, b_0, ab_{00}, z_0\rangle \rightarrow \langle a_1, b_0, ab_{00}, z_0\rangle \rightarrow \langle a_1, b_0, ab_{10}, z_0\rangle \rightarrow \langle a_0, b_0, ab_{10}, z_0\rangle \rightarrow \langle a_0, b_1, ab_{10}, z_0\rangle$$

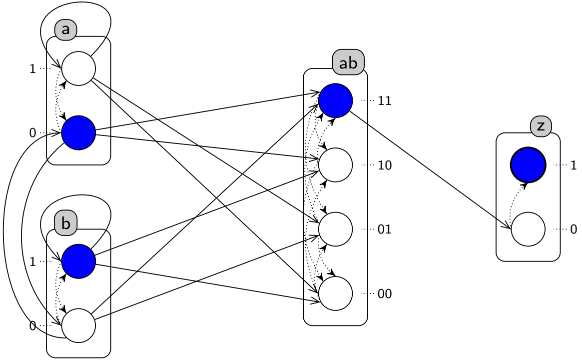
Adapting the expressivity of PH



Drawback: Cooperations are too "loose" to be as expressive as ADN.

$$\langle a_0, b_0, ab_{00}, z_0\rangle \rightarrow \langle a_1, b_0, ab_{00}, z_0\rangle \rightarrow \langle a_1, b_0, ab_{10}, z_0\rangle \rightarrow \langle a_0, b_0, ab_{10}, z_0\rangle$$
$$\rightarrow \langle a_0, b_1, ab_{10}, z_0\rangle \rightarrow \langle a_0, b_1, ab_{11}, z_0\rangle$$

Adapting the expressivity of PH



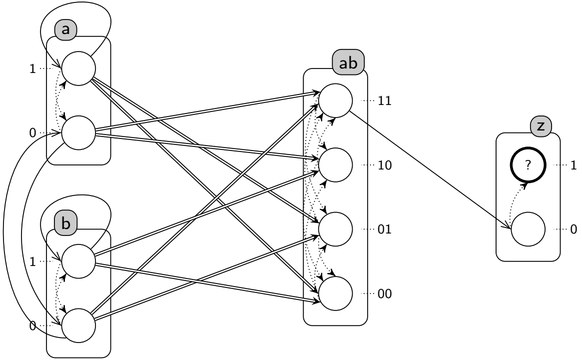
Drawback: Cooperations are too "loose" to be as expressive as ADN.

$$\langle a_0, b_0, ab_{00}, z_0\rangle \rightarrow \langle a_1, b_0, ab_{00}, z_0\rangle \rightarrow \langle a_1, b_0, ab_{10}, z_0\rangle \rightarrow \langle a_0, b_0, ab_{10}, z_0\rangle \rightarrow \langle a_0, b_1, ab_{10}, z_0\rangle \rightarrow \langle a_0, b_1, {\color{red}ab_{11}}, z_0\rangle \rightarrow \langle a_0, b_1, {\color{red}ab_{11}}, {\color{red}z_1}\rangle$$

The cooperativity should be: $a_1 \wedge b_1$ **simultaneously** *i.e.* "in the same state"

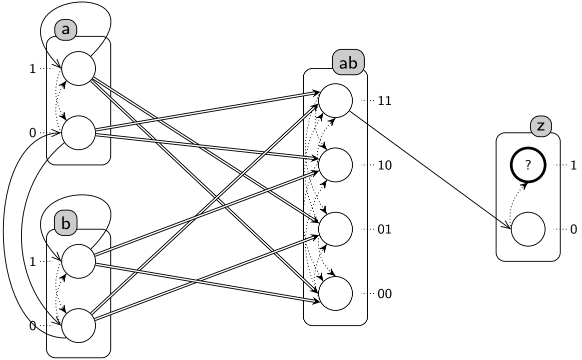
but the model behaves like: $\mathbf{P}(a_1) \wedge \mathbf{P}(b_1)$ with $\mathbf{P}$ = "previously"

Adapting the expressivity of PH



- Prioritise actions updating cooperative sorts (non-biological actions)
- All other actions remain unprioritised (evolutions with delays)

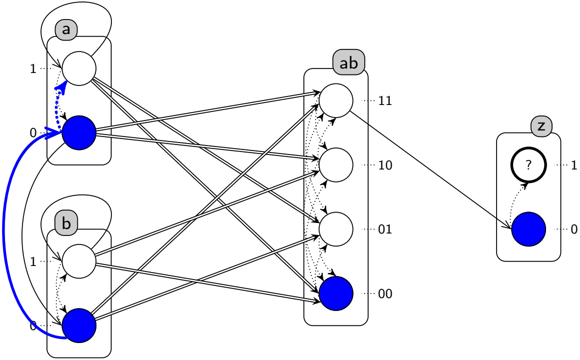
Adapting the expressivity of PH



- Prioritise actions updating cooperative sorts (non-biological actions)
- All other actions remain unprioritised (evolutions with delays)

$\Rightarrow$ Whenever a regular action is played, all cooperative sorts are already updated

Adapting the expressivity of PH

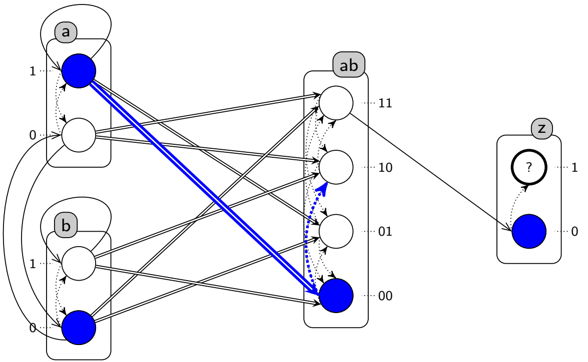


- Prioritise actions updating cooperative sorts (non-biological actions)
- All other actions remain unprioritised (evolutions with delays)

$\Rightarrow$ Whenever a regular action is played, all cooperative sorts are already updated

$\langle a_0, b_0, ab_{00}, z_0 \rangle$

Adapting the expressivity of PH

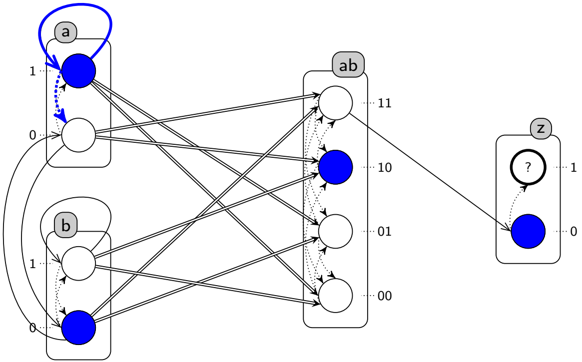


- Prioritise actions updating cooperative sorts (non-biological actions)
- All other actions remain unprioritised (evolutions with delays)

$\Rightarrow$ Whenever a regular action is played, all cooperative sorts are already updated

$\langle a_0, b_0, ab_{00}, z_0\rangle \rightarrow \langle a_1, b_0, ab_{00}, z_0\rangle$

Adapting the expressivity of PH

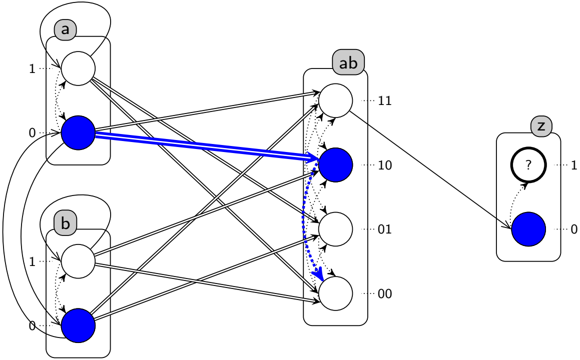


- Prioritise actions updating cooperative sorts (non-biological actions)
- All other actions remain unprioritised (evolutions with delays)

$\Rightarrow$ Whenever a regular action is played, all cooperative sorts are already updated

$\langle a_0, b_0, ab_{00}, z_0\rangle \rightarrow \langle a_1, b_0, ab_{00}, z_0\rangle \rightarrow \langle a_1, b_0, ab_{10}, z_0\rangle$

Adapting the expressivity of PH

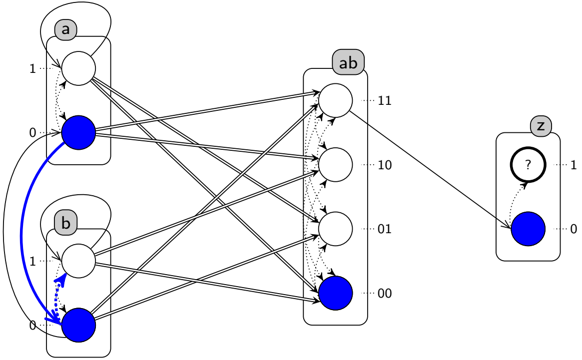


- Prioritise actions updating cooperative sorts (non-biological actions)
- All other actions remain unprioritised (evolutions with delays)

$\Rightarrow$ Whenever a regular action is played, all cooperative sorts are already updated

$\langle a_0, b_0, ab_{00}, z_0 \rangle \rightarrow \langle a_1, b_0, ab_{00}, z_0 \rangle \rightarrow \langle a_1, b_0, ab_{10}, z_0 \rangle \rightarrow \langle a_0, b_0, ab_{10}, z_0 \rangle$

Adapting the expressivity of PH

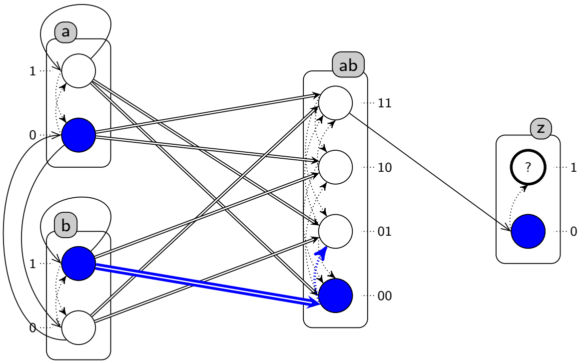


- Prioritise actions updating cooperative sorts (non-biological actions)
- All other actions remain unprioritised (evolutions with delays)

$\Rightarrow$ Whenever a regular action is played, all cooperative sorts are already updated

$\langle a_0, b_0, ab_{00}, z_0\rangle \rightarrow \langle a_1, b_0, ab_{00}, z_0\rangle \rightarrow \langle a_1, b_0, ab_{10}, z_0\rangle \rightarrow \langle a_0, b_0, ab_{10}, z_0\rangle$
$\rightarrow \langle a_0, b_0, ab_{00}, z_0\rangle$

Adapting the expressivity of PH

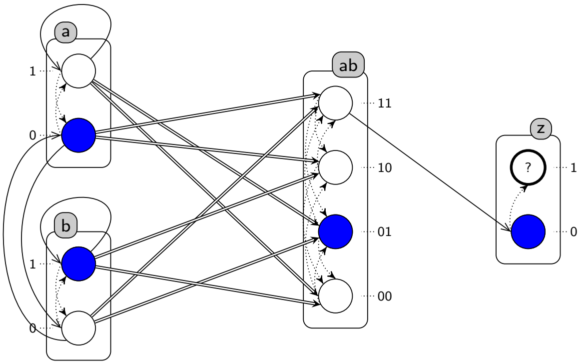


- Prioritise actions updating cooperative sorts (non-biological actions)
- All other actions remain unprioritised (evolutions with delays)

$\Rightarrow$ Whenever a regular action is played, all cooperative sorts are already updated

$$\langle a_0, b_0, ab_{00}, z_0\rangle \rightarrow \langle a_1, b_0, ab_{00}, z_0\rangle \rightarrow \langle a_1, b_0, ab_{10}, z_0\rangle \rightarrow \langle a_0, b_0, ab_{10}, z_0\rangle$$
$$\rightarrow \langle a_0, b_0, ab_{00}, z_0\rangle \rightarrow \langle a_0, b_1, ab_{00}, z_0\rangle$$

Adapting the expressivity of PH



- Prioritise actions updating cooperative sorts (non-biological actions)
- All other actions remain unprioritised (evolutions with delays)

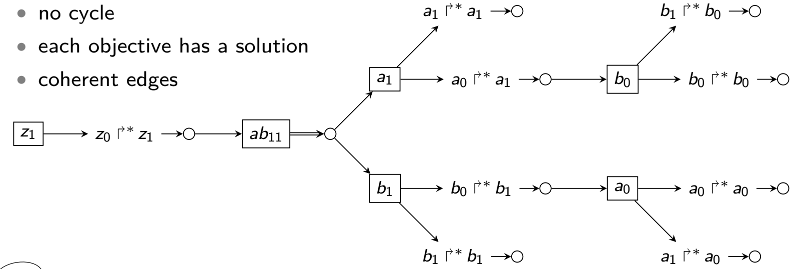
$\Rightarrow$ Whenever a regular action is played, all cooperative sorts are already updated

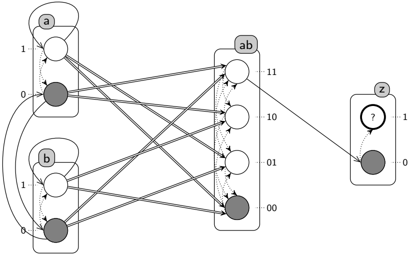
$$\langle a_0, b_0, ab_{00}, z_0\rangle \rightarrow \langle a_1, b_0, ab_{00}, z_0\rangle \rightarrow \langle a_1, b_0, ab_{10}, z_0\rangle \rightarrow \langle a_0, b_0, ab_{10}, z_0\rangle$$
$$\rightarrow \langle a_0, b_0, ab_{00}, z_0\rangle \rightarrow \langle a_0, b_1, ab_{00}, z_0\rangle \rightarrow \langle a_0, b_1, ab_{01}, z_0\rangle$$

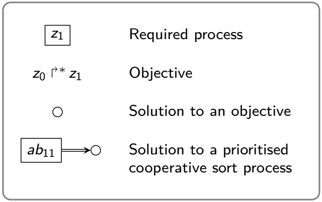
Static analysis with prioritised actions

Sufficient condition:

- no cycle
- each objective has a solution
- coherent edges



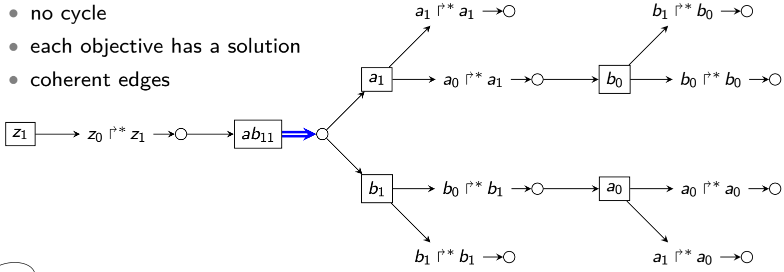


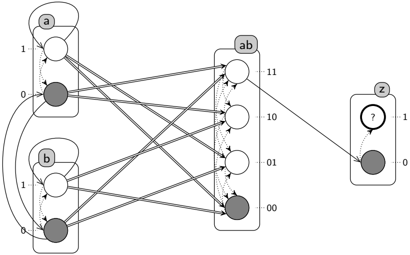


Static analysis with prioritised actions

Sufficient condition:

- no cycle
- each objective has a solution
- coherent edges



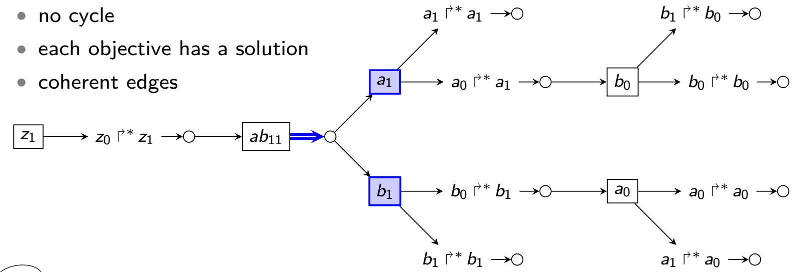


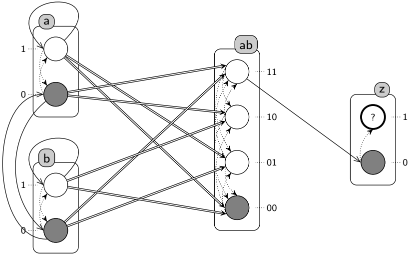
$\boxed{z_1}$	Required process
$z_0 \Rsh^* z_1$	Objective
○	Solution to an objective
$\boxed{ab_{11}} \Longrightarrow$ ○	Solution to a prioritised cooperative sort process

Static analysis with prioritised actions

Sufficient condition:

- no cycle
- each objective has a solution
- coherent edges



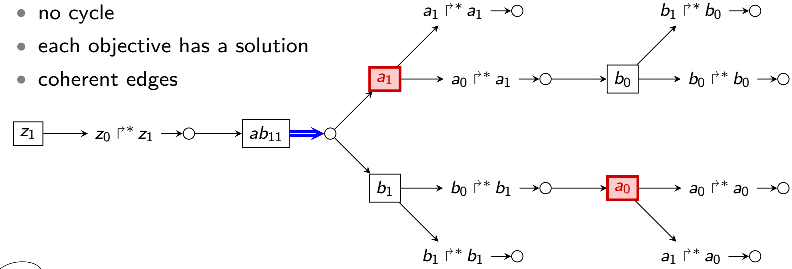


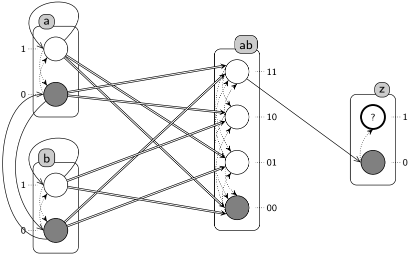
$\boxed{z_1}$	Required process
$z_0 \Rsh^* z_1$	Objective
○	Solution to an objective
$\boxed{ab_{11}} \Rightarrow$ ○	Solution to a prioritised cooperative sort process

Static analysis with prioritised actions

Sufficient condition:

- no cycle
- each objective has a solution
- coherent edges



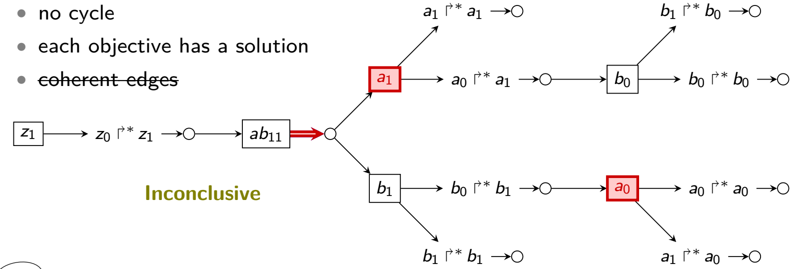


z_1	Required process
$z_0 \upharpoonright^* z_1$	Objective
○	Solution to an objective
$ab_{11} \Rightarrow$ ○	Solution to a prioritised cooperative sort process

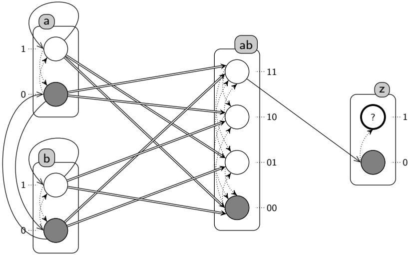
Static analysis with prioritised actions

Sufficient condition:

- no cycle
- each objective has a solution
- ~~coherent edges~~



Inconclusive



$\boxed{z_1}$	Required process
$z_0 \upharpoonright^* z_1$	Objective
○	Solution to an objective
$\boxed{ab_{11}} \Rightarrow$ ○	Solution to a prioritised cooperative sort process

Implementation

Complexity:

- Building the graph:
  - Polynomial in the number of sorts
  - Exponential in the number of processes in each sort
- Analysing the graph:
  - Polynomial in the size of the graph

Implementation

Complexity:

- Building the graph:
  - Polynomial in the number of sorts
  - Exponential in the number of processes in each sort
- Analysing the graph:
  - Polynomial in the size of the graph

Model	Sorts	Procs	Actions	States	libddd[1]	GINsim[2]	PINT
egfr20	35	196	670	2^{64}		<1s	**0.35s**
tcrsig40	54	156	301	2^{73}		∞	**0.2s**
tcrsig94	133	448	1124	2^{194}	[13min – ∞]		**0.8s**

[1] LIP6/Move
[2] TAGC/IGC

egfr20: [Epidermal Growth Factor Receptor, by Özgür Sahin *et al.*]
tcrsig40: [T-Cell Receptor Signaling, by Steffen Klamt *et al.*]
tcrsig94: [T-Cell Receptor Signaling, by Julio Saez-Rodriguez *et al.*]

Summary

- The Process Hitting framework
  - $\rightarrow$ Restricted concurrent actions
  - $\rightarrow$ Efficient static analysis on biological models (few expression levels)
- But raw Process Hitting is insufficient to models ADNs
  - $\rightarrow$ How to represent cooperations?
  - $\rightarrow$ Cooperative sorts only represent a combination of past states
- Solution: prioritised actions
  - $\rightarrow$ Accurate cooperative sorts
  - $\rightarrow$ Expressivity of ADN is reached

Conclusion

- Achieved:
  - Rise the expressivity of PH
  - Efficient reachability analysis in ADNs
- Value:
  - Model a whole class of ADNs in one PH model
  - Efficiently analyse reachability for the whole class
  - Refine the PH model to match desired behaviour
  - Infer the underlying class of ADNs
    [Folschette, Paulevé, Inoue, Magnin, Roux in *Computational Methods in Systems Biology*, 2012]

Conclusion

- Achieved:
  - Rise the expressivity of PH
  - Efficient reachability analysis in ADNs
- Value:
  - Model a whole class of ADNs in one PH model
  - Efficiently analyse reachability for the whole class
  - Refine the PH model to match desired behaviour
  - Infer the underlying class of ADNs
    [Folschette, Paulevé, Inoue, Magnin, Roux in *Computational Methods in Systems Biology*, 2012]

Outlook

- Allow prioritised actions even for biological evolutions
- Allow $n > 2$ classes of priority

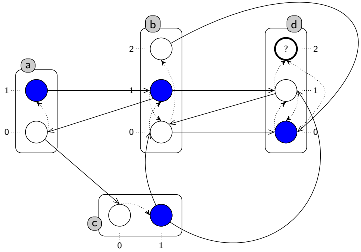
  $\rightarrow$ Model actions with delays by using priorities

Bibliography

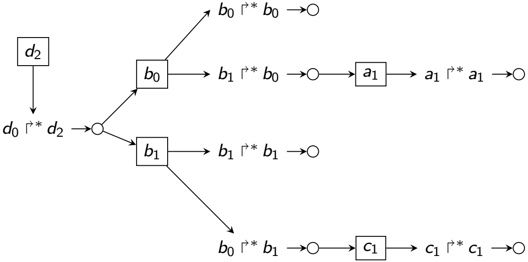
- Loïc Paulevé, Morgan Magnin, Olivier Roux. Refining dynamics of gene regulatory networks in a stochastic π-calculus framework. In Corrado Priami, Ralph-Johan Back, Ion Petre, and Erik de Vink, editors: *Transactions on Computational Systems Biology XIII*, Lecture Notes in Computer Science, 171-191. Springer Berlin Heidelberg, 2011.
- Loïc Paulevé, Morgan Magnin, Olivier Roux. Static analysis of biological regulatory networks dynamics using abstract interpretation. *Mathematical Structures in Computer Science*. 2012.
- Hidde de Jong. Modeling and simulation of genetic regulatory systems: a literature review, *Journal of Computational biology* 9(1), 67–103. 2002.
- Adrien Richard and Jean-Paul Comet. Necessary conditions for multistationarity in discrete dynamical systems. *Discrete Applied Mathematics* 155(18), 2403–2413. 2007.
- Élisabeth Remy, Paul Ruet and Denis Thieffry. Graphic requirements for multistability and attractive cycles in a boolean dynamical framework, *Advances in Applied Mathematics* 41(3), 335-350. Elsevier, 2008.
- Gilles Bernot, Jean-Paul Comet, Adrien Richard and Janine Guespin. Application of formal methods to biological regulatory networks: extending Thomas' asynchronous logical approach with temporal logic, *Journal of Theoretical Biology*, 229(3), 339–347. Elsevier, 2004.
- Sohei Ito, Naoko Izumi, Shigeki Hagihara and Naoki Yonezaki. Qualitative analysis of gene regulatory networks by satisfiability checking of Linear Temporal Logic, in 2010 IEEE International Conference on *BioInformatics and BioEngineering*, 232–237. IEEE, 2010.
- Maxime Folschette, Loïc Paulevé, Katsumi Inoue, Morgan Magnin, Olivier Roux. Concretizing the Process Hitting into Biological Regulatory Networks. In David Gilbert and Monika Heiner, editors, *Computational Methods in Systems Biology X*, Lecture Notes in Computer Science, 166–186. Springer Berlin Heidelberg, 2012.

Thank you

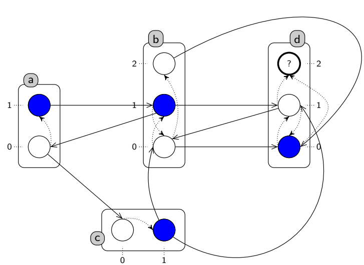
Under-approximation



$\boxed{d_2}$	Required process
$d_0 \Rsh^* d_2$	Objective
○	Solution to an objective

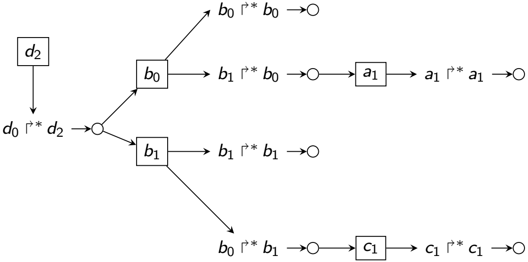


Under-approximation



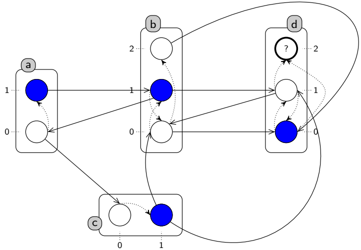
Sufficient condition:

- no cycle
- each objective has a solution



$\boxed{d_2}$	Required process
$d_0 \Rsh^* d_2$	Objective
○	Solution to an objective

Under-approximation

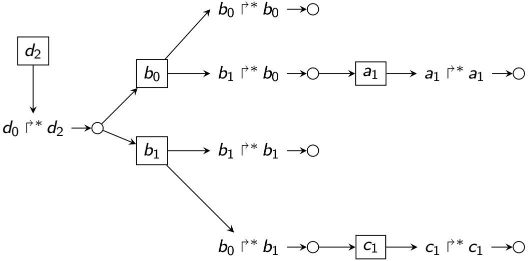


Sufficient condition:

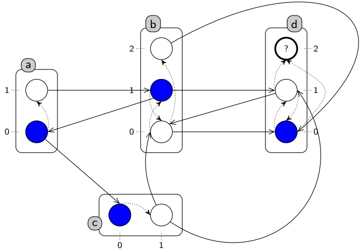
- no cycle
- each objective has a solution

R is **true**

$\boxed{d_2}$	Required process
$d_0 \upharpoonright^* d_2$	Objective
○	Solution to an objective

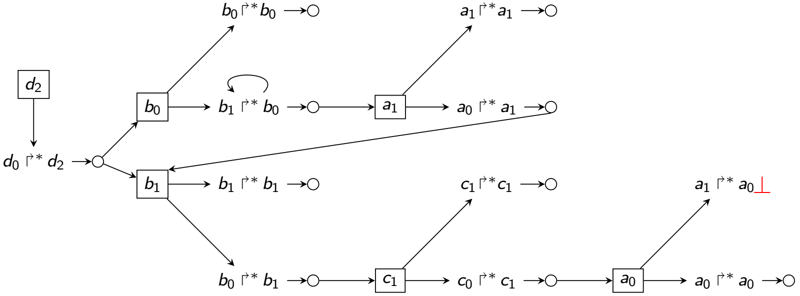


Under-approximation

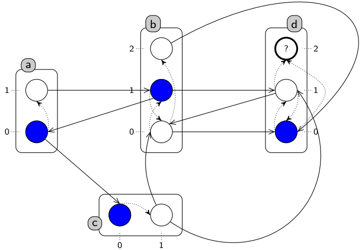


Sufficient condition:

- no cycle
- ~~each objective has a solution~~



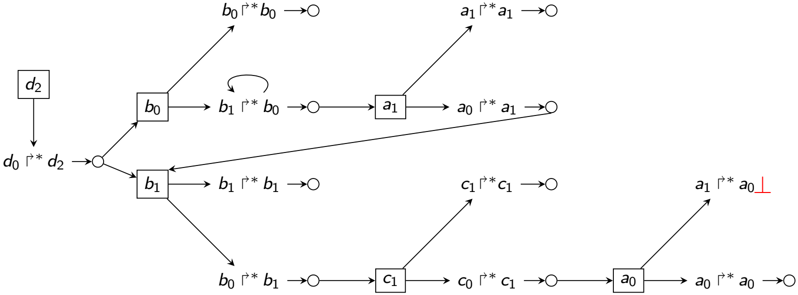
Under-approximation



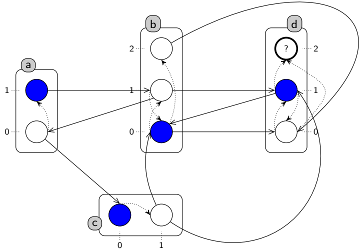
Sufficient condition:

- no cycle
- ~~each objective has a solution~~

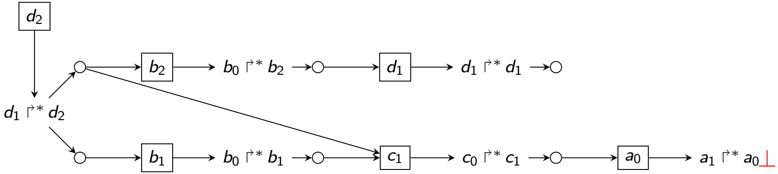
Inconclusive



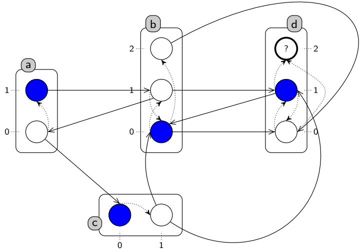
Over-approximation



Necessary condition:



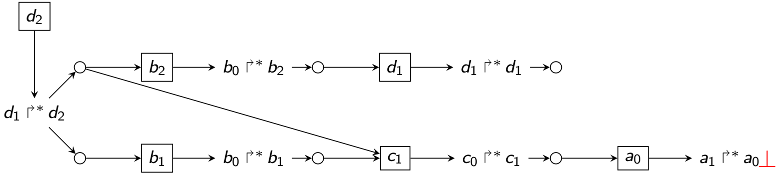
Over-approximation



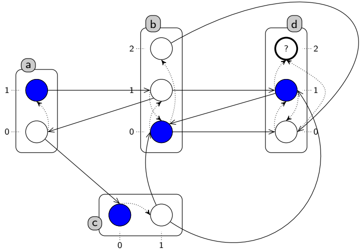
Necessary condition:

There exists a traversal with no cycle

- objective → follow **one** solution
- solution → follow **all** processes
- process → follow **all** objectives



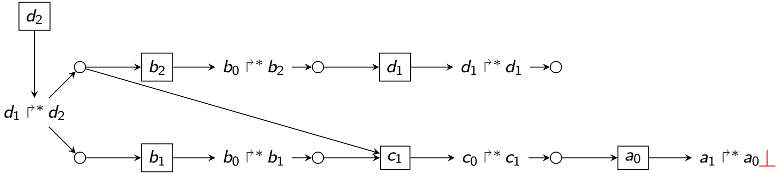
Over-approximation



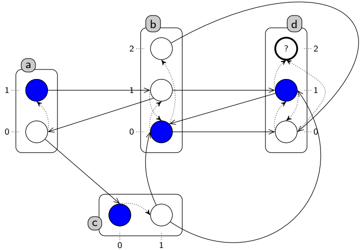
Necessary condition:

~~There exists a traversal~~ with no cycle

- ~~objective → follow **one** solution~~
- solution → follow **all** processes
- process → follow **all** objectives



Over-approximation

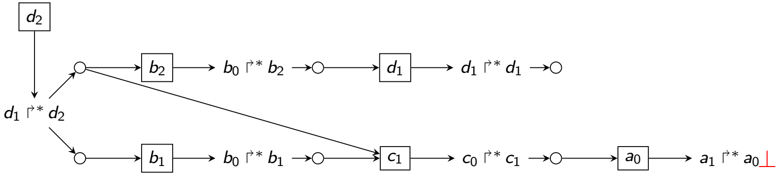


Necessary condition:

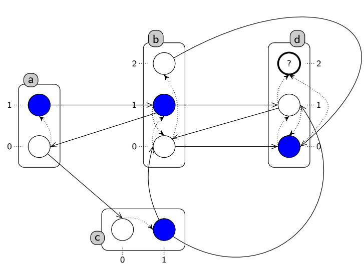
~~There exists a traversal~~ with no cycle

- ~~objective → follow **one** solution~~
- solution → follow **all** processes
- process → follow **all** objectives

R is **false**



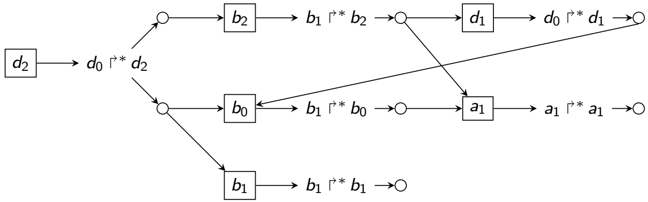
Over-approximation



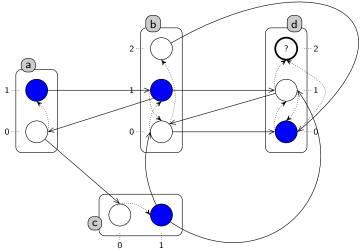
Necessary condition:

There exists a traversal with no cycle

- objective → follow **one** solution
- solution → follow **all** processes
- process → follow **all** objectives



Over-approximation

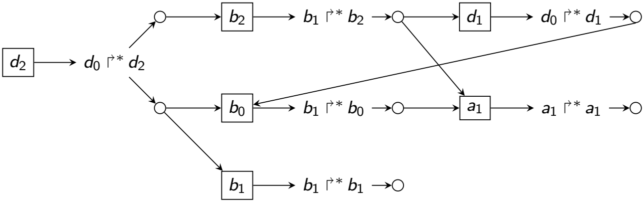


Necessary condition:

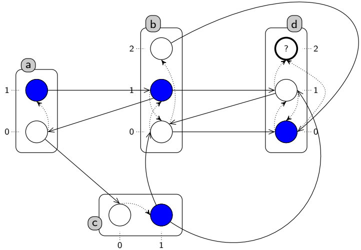
There exists a traversal with no cycle

- objective → follow **one** solution
- solution → follow **all** processes
- process → follow **all** objectives

Inconclusive



Over-approximation



Necessary condition:

There exists a traversal with no cycle

- objective → follow **one** solution
- solution → follow **all** processes
- process → follow **all** objectives

Inconclusive